AIAA-2001-1625

MULTILEVEL PARALLEL OPTIMIZATION USING MASSIVELY PARALLEL STRUCTURAL DYNAMICS

Bart G. van Bloemen Waanders[*], Michael S. Eldred[‡], Anthony A. Giunta[†],
Garth M. Reese, Manoj K. Bhardwaj[*], Clay W. Fulcher

Sandia National Laboratories[‡], Albuquerque, NM 87185 USA

Abstract

A large scale optimization of an electronics package has been completed using a massively parallel structural dynamics code. The optimization goals were to maximize safety margins for stress and acceleration resulting from transient impulse loads, while remaining within strict mass limits. The optimization process utilized nongradient, gradient, and approximate optimization methods in succession to modify shell thickness and foam density values within the electronics package. This combination of optimization methods was successful in improving the performance from an infeasible design which violated stress allowables by a factor of two to a completely feasible design with positive design margins, while remaining within the mass limits. In addition, a tradeoff curve of mass versus safety margin was developed to facilitate the design decision process. These studies employed the ASCI Red supercomputer and utilized multiple levels of parallelism on up to 2560 processors. In one portion of this optimization study, a series of calculations were performed on ASCI Red in four days, where an equivalent calculation on a single desktop computer would have taken greater than 10 years to complete.

1 Introduction

This report describes the design optimization of an electronics package (EP) which is one component of a re-entry vehicle (RV). The design study was performed using massively parallel, high-fidelity structural dynamics simulations conducted on the Accelerated Strategic Computing Initiative (ASCI) Red supercomputer at Sandia National Laboratories.

Optimization problems of this complexity and computational expense pose many technical challenges. For good computational efficiency, the structural dynamics and optimization codes must be scalable to a large numbers of processors (order $10^2 - 10^4$). For the structural dynamics software, this entails the use of specific mathematical techniques (e.g., linear solvers) that exploit both the structure of the finite element model and the hardware configuration of the parallel computer. For the optimization software, parallel scheduling of simulations must also exploit the hardware configuration of the parallel computer, and the optimization methods must be robust to nonsmooth response values generated in the structural dynamics simulation.

Over the course of this study, several optimization methods were applied in an effort to improve the design of the EP. The results of this study demonstrate the utility of having a "toolbox" of sensitivity analysis and optimization algorithms for performing engineering design optimization studies.

Details of this design effort are given below, with Sections 2-5 providing background information on the electronics package model, the Salinas structural dynamics software, the DAKOTA optimization toolkit, and the ASCI Red supercomputer, respectively. Section 6 describes the formulations, methods, and results in the EP design optimization problem, and Section 7 provides concluding remarks.

2 Electronics Package Model

The motivation for the optimization study was to help designers improve the structural integrity of a new EP structural design concept. Since this EP design is a refurbish-

[*] Member, AIAA

[†] Senior Member, AIAA

[‡] Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

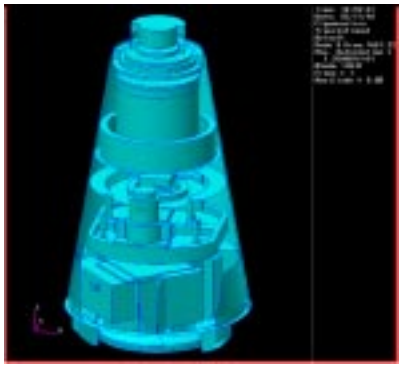

Figure 1: A CAD model of the electronics package.

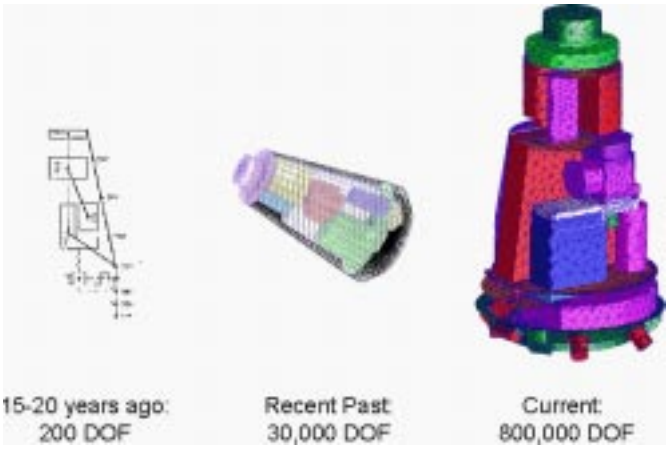

Figure 2: Historical progression of the electronics package finite element model fidelity.

ment for the RV, it provided the opportunity to incorporate several new components into the existing package. However, an important requirement was to avoid changing the flight characteristics of the RV, so a restriction of no more than 10% deviation from the nominal EP mass was imposed. In order to add functionality and maintain mass, the EP design concept replaced some structure with support foam. Thus, the design problem is a challenging one in that an EP design concept with less structural support must still survive high stresses and accelerations from severe RV structural loading conditions. A solid model of the EP design concept is shown in Figure 1.

Over time, the level of fidelity in structural dynamics analysis has increased significantly (Figure 2) as a result of more advanced computers and, most recently, the availability of a massively parallel structural dynamics code. In this study, the model was discretized using a finite element model having 500,000 degrees of freedom. This model captures the salient features of the EP in sufficient detail for the optimization study. However, models of greater than 10 million degrees of freedom have been used to resolve additional detail in the EP. The EP finite element model was decomposed into 256 subdomains for parallel processing. During the finite element analysis, each subdomain was associated with a single processor, and the data input/output (I/O) operations were each spread to 256 separate, subdomain specific files.

For this design study, 55 finite element blocks were identified as the most critical. Some of these blocks were structural shell elements within the EP, while others were blocks of foam encapsulant used between and around the EP components. The design variables for this study were the shell thicknesses of a subset of the structural blocks, and the density values for a subset of the foam encapsulant blocks. These subsets were selected based on a modal sensitivity analysis, with those blocks having the largest impact on the first 100 frequencies (greatest number of frequency derivatives greater than a threshold) being selected as design parameters. Stress and acceleration values within each of the 55 blocks served as response quantities to be used during the optimization process.

3 Salinas: Massively Parallel Structural Dynamics

Salinas [1] is a general-purpose, finite element structural dynamics code designed to be scalable on massively parallel computers. Currently the code offers static analysis, direct implicit transient analysis, eigenvalue analysis for computing modal response, and modal superposition-based frequency response and transient response. In addition, semi-analytical derivatives of many response quantities with respect to user-selected design parameters are calculated. Salinas also includes an extensive library of standard one-, two-, and three-dimensional elements, nodal and element loading, and multi-point constraints. Salinas solves systems of equations using an iterative, multilevel solver, which is specifically designed to exploit massively parallel machines.

Salinas uses a linear solver that was selected based on the criteria of robustness, accuracy, scalability and efficiency. Direct methods based on sparse Gaussian elimination were considered but they appear to provide poor robustness on platforms, such as ASCI Red, with fast communication, many processors, and limited per processor memory. General purpose iterative solvers, such as

the implementation of the preconditioned conjugate gradient method with over-lapping Schwartz preconditioner available in Aztec[2], were also evaluated. These methods converged too slowly for the linear systems obtained from the discretization of structures by high order plate and shell elements. In this case, the underlying partial differential equation is the fourth order biharmonic equation for which special purpose iterative solvers are necessary. This led to the selection of a multilevel domain decomposition method, Finite Element Tearing and Interconnect (FETI)[3], that is the most successful parallel solver known to the authors for the linear systems applicable to structural mechanics. FETI is a mature solver, with some versions used in commercial finite element packages such as ANSYS[4]. For plates and shells, the singularity in the linear systems has been traced to the subdomain corners. To solve such linear systems, an additional coarse problem is automatically introduced that removes the corner point singularity. FETI is scalable in the sense that as the number of unknowns increases and the number of unknowns per processor remains constant, the time to solution does not increase. Further, FETI is accurate in the sense that the convergence rate does not deteriorate as the iterates converge. Finally the computational bottleneck in FETI, a sparse direct subdomain solve, is amenable to high performance solution methods.

An eigensolver was selected for Salinas based on the same criteria; robustness, accuracy, scalability and efficiency. Both a Lanczos-based solver[5] and subspace iteration were evaluated. The Lanczos algorithm solves the minimal number of linear systems required to approximate a set of modes to a given accuracy, and Lanczos-based methods are significantly more efficient than subspace iteration. Subspace iteration is a comparatively simple algorithm that is believed to be somewhat less sensitive to linear solver accuracy than Lanczos-based methods. Structural models are known for which the FETI solver does not converge, but in these cases the accuracy is too low for either subspace iteration or Lanczos-based methods to compute accurate modes. The PARPACK Lanczos-based solver was selected because the memory usage is minimal, the software is reliable, and the number of linear systems solved per mode is nearly minimized. PARPACK[6] is scalable and achieves BLAS2 performance.

4 DAKOTA: Multilevel Parallel Optimization

The DAKOTA toolkit [7] is a software framework for systems analysis, encompassing optimization, parameter estimation, uncertainty quantification, design of computer experiments, and sensitivity analysis. It provides generic simulation interfacing facilities which allow the use of a variety of engineering and physics simulation codes as function evaluators within an iterative loop. DAKOTA manages the complexities of its analysis and optimization capabilities through the use of object-oriented abstraction, class hierarchies, and polymorphism. A variety of optimization algorithms are available, ranging from gradient-based nonlinear programming methods to nongradient-based pattern search methods. The flexibility of the framework allows for easy incorporation of the latest external and internal algorithmic developments. In addition, the variety of methods and interfaces can be used as building blocks for more sophisticated studies, such as surrogate-based optimization, hybrid optimization, mixed integer nonlinear programming, and optimization under uncertainty.

Parallelism is an essential component of the DAKOTA framework. Particular emphasis has been given to simultaneously exploiting parallelism at a variety of levels in order of achieve near-linear scaling on massively parallel computers. For example, DAKOTA can manage concurrent optimizations, each with concurrent function evaluations, each with concurrent analyses that utilize multiple processors. Reference [8] provides guidance on how to select partitioning schemes and scheduling algorithms within these levels in order to maximize overall parallel efficiency and to ensure robustness with respect to variabilities (e.g., simulation duration variability). A common case is two levels of parallelism, in which concurrent function evaluations each run on multiple processors. In this study, DAKOTA employed two levels of parallelism by managing up to 10 concurrent Salinas invocations, each of which required 256 compute nodes. Through this combination of coarse-grained and fine-grained parallel computing, DAKOTA was able to effectively utilize 2560 processors and achieve rapid turnaround on this large-scale design study.

5 ASCI Red Supercomputer

5.1 Architecture Review

For this optimization study, substantial compute resources were required. Within Sandia National Laboratories, one of the primary production computing platforms is the ASCI Red supercomputer [9, 10]. ASCI Red is a massively parallel, message-passing, multiple input multiple data (MIMD) computer. It achieves multiple TeraFLOPS (trillion floating point operations per second) peak performance. It is designed so that file I/O, memory, disk capacity, and communication are scalable. Standard parallel programming libraries, such as the Message Passing Interface (MPI) [11] make it relatively simple to port parallel applications to this system.

Table 1: Hardware and performance characteristics of the ASCI Red supercomputer.

Service Nodes	16
Compute Nodes	4640
Total Processors	9536
System RAM (TeraBytes)	1.21
Compute Node Peak Performance (MegaFLOPS)	666
System Peak Performance (TeraFLOPS)	3.15

The processors in the ASCI Red supercomputer are organized into four partitions: compute, service, system, and I/O. Of these, the service partition provides an integrated, scalable host that supports interactive users, application development, and system administration. This partition runs a full UNIX operating system. The parallel applications execute in the compute partition, which contains nodes optimized for floating point performance and for high bandwidth communication. This partition executes the Cougar operating system [12] which is a lightweight kernel intended to leave as much node memory as possible for the application. Each compute node consists of two 333 MHz Intel Pentium-II Xeon Core processors with 256 Mbytes of random access memory (RAM). In this study, only one processor per node was used for computation while the other processor was used for communication, although a new "virtual node" capability allows the use of both node processors for computation. The system hardware and performance attributes of ASCI Red are summarized in Table 1.

5.2 Salinas/DAKOTA Implementation on ASCI Red

DAKOTA can be interfaced with simulation codes in a variety of ways depending on the level of intrusiveness one is willing to support, on the desired performance, and on the underlying compute architecture. The simplest approach is the UNIX "system call" method. This is the least intrusive method in that the simulation can be used as is, with no modifications. It is also the least efficient method in that it incurs the overhead of creating separate processes for the simulations. In practice, this overhead is usually small relative to the expense of the simulations. The most computationally efficient interface technique is the "direct" method in which the simulation code (e.g., Salinas) is linked into DAKOTA as a callable function. While this direct interface is efficient, it is intrusive to the simulation code since the code must be transformed to a subroutine and, in the parallel case, made modular on an MPI communicator. In addition, it complicates the use of pre- and post-processing tools (e.g., mesh generation, domain decomposition and reconstitution).

These two interfacing approaches have additional distinctions when applied on massively parallel computers which employ a service/compute node distinction. In particular, the system call approach involves the execution of DAKOTA on the service nodes where it creates concurrent simulation driver processes on the service nodes. Each of these simulation drivers then launches a parallel simulation into the compute node partition. DAKOTA must then continuously monitor for the completion of these simulations, again utilizing service node compute resources. The direct approach, on the other hand, involves the execution of a combined executable on the compute nodes only. The management of concurrent multiprocessor simulations is performed internally using MPI communicators. Clearly, the system call approach places far greater demands on the service partition than the direct approach.

For this study, the UNIX system call interface method was used which allowed the use of a separate, unmodified Salinas executable. In this case, DAKOTA was run on the service node partition where it coordinated concurrent Salinas jobs in the compute partition. This is depicted in Figure 3. Not shown in this figure are the pre- and post-processing steps needed to communicate between DAKOTA and Salinas. Values of the design variables were written by DAKOTA to a file and then incorporated into the Salinas input file using a Sandia-developed file parsing program. The output of Salinas was post-processed to provide the mass and safety margin data values needed for the optimization methods in DAKOTA. While the results from each simulation could be gathered into a single file for post-processing, it was more expedient to evaluate safety margins across separate subdomain databases. This entire cycle was automated using a single driver script that was invoked by DAKOTA. While DAKOTA was executed on a single service node and each of the system calls to concurrent Salinas drivers were initiated from this single service node, a resident load spreading utility would relocate Salinas monitoring processes among the entire service partition.

5.3 Computational Issues

Optimization studies using nonintrusive interfacing approaches impose different loads on supercomputers in comparison to single parallel executables. In particular, multiple concurrent jobs put much higher load on the service nodes than single jobs. The service nodes on ASCI Red were designed to manage basic coordination tasks and not significant floating point operations or heavy I/O demands. Although the individual processors are capable of computations, there are simply not enough service nodes to sustain significant activities. In the case of this

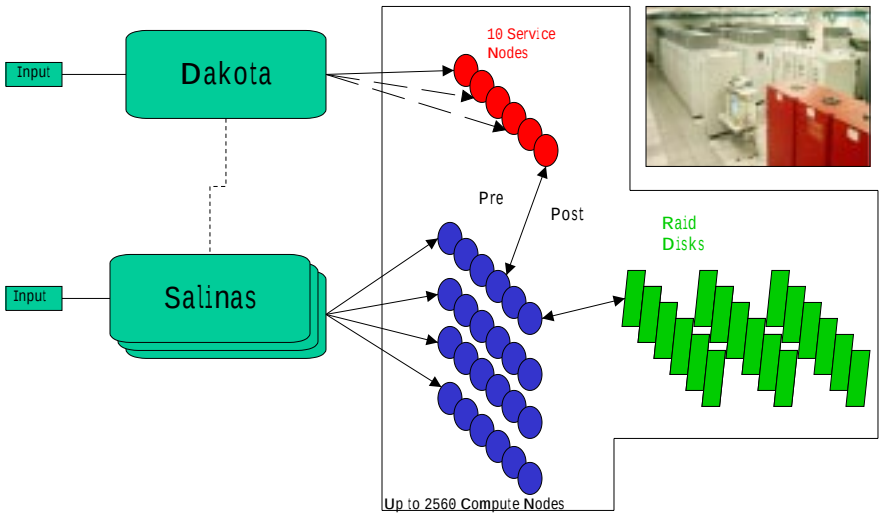

Figure 3: A depiction of the DAKOTA/Salinas implementation on ASCI Red.

design study, the service nodes were responsible for managing the optimization process (running the optimizer and querying for job completion) as well as concurrent simulation driver processes and multiprocessor simulation initiation processes. At certain points during the studies, Salinas jobs would hang on initiation. In the worst of these incidents, a service node became overloaded and crashed, which necessitated a full reboot of the machine.

It appears as though the observed reliability problems stemmed not directly from the total amount of work being performed, but rather the closely synchronized nature of concurrent simulation invocation. In this study, it was found that staggering the Salinas job initiations by a few seconds allowed the load spreading utility sufficient time to spread the Salinas monitoring processes among the service nodes, which resulted in improved reliability. A continuing effort is focusing on improving the robustness of the service nodes during optimization procedures.

6 Optimization Results

The overall objective of this optimization study was to modify the electronics package so that the new design satisfied safety margin requirements and remained within a strict mass budget. Four different optimization algorithms and two different optimization problem formulations were applied to this EP design problem.

6.1 Phase 1: Nongradient and Gradient-Based Optimization

6.1.1 Coordinate Pattern Search Algorithm

The initial optimization strategy for the EP redesign was to maximize the minimum safety margin (SM), subject to constraints on the EP mass. Four shell thickness parameters and one foam density parameter used in the EP model were selected as design variables for this optimization case. These five parameters were the most sensitive based on the results of a modal sensitivity study.

The Phase 1 optimization problem was formulated in DAKOTA as follows:

$$\begin{array}{ll} \text{maximize} & SM_{min} \\ \text{subject to} & 0.9M_{nom} \leq M \leq 1.1M_{nom}, \\ & (x_n)_{LB} \leq x_n \leq (x_n)_{UB}, \end{array} \quad (1)$$

where SM_{min} is the minimum over all safety margin values, M is the mass of the EP, M_{nom} is the nominal mass of the EP, and x_n is the vector of design variables with lower and upper bounds $(x_n)_{LB}$ and $(x_n)_{UB}$, respectively, for $n = 1, \ldots, 5$. The safety margin values were computed for the EP internal components based on either a stress allowable value or an acceleration allowable value. The safety margins based on stress values were computed as

$$SM_i = \frac{\sigma_i^a}{\sigma_i} - 1, \text{for } i = 1, \ldots, 42, \quad (2)$$

where σ^a is the allowable stress and σ_i is the computed maximum stress in the i^{th} block. Similarly, the safety margins on acceleration values were computed as:

$$SM_i = \frac{g_i^a}{g_i} - 1, \text{for } i = 43, \ldots, 55, \tag{3}$$

where g^a is the allowable acceleration level, and g_i is the computed maximum acceleration level in the i^{th} block.

In these SM definitions (Equations 2 and 3), the fractional term is called the safety factor. Here, σ_i^a is the yield stress for the particular material block and g_i^a was fixed at a constant value for all relevant material blocks. The nominal EP design had $SM_{min} = -0.48$, which can be interpreted as some part of the EP being exposed to twice the allowable stress/acceleration and obviously prone to failure.

Since this Phase 1 optimization formulation was expected to be nonsmooth due to switching among various components with the lowest safety margin, a nongradient-based method was selected for the initial optimization of the EP. This method was the coordinate pattern search method (CPS) [13] contained in the Stochastic Global Optimization (SGOPT) [14] software package (SGOPT is linked into the DAKOTA toolkit).

A single Salinas function evaluation required approximately 40 minutes on 256 processors of ASCI Red. Using the two-level parallel capabilities in DAKOTA, 10 instances of Salinas were executed concurrently. This completed a full optimization cycle of the CPS algorithm in one pass since CPS requires $2n$ function evaluations on each cycle (i.e., 10 Salinas jobs performed concurrently for $n = 5$). The CPS method was able to improve the minimum safety margin from the nominal value of -0.48 to -0.21 with a mass increase of 5.4%, using a total of 171 function evaluations (Table 2). The pattern search made good progress until three separate margin functions were active at the current optimum. This occurrence adversely affected the convergence rate of the pattern search method, as it was difficult to generate a step which simultaneously improved all three safety margins from the finite set of coordinate search directions.

6.1.2 NPSOL-SQP Algorithm

At this stage of the optimization, it was clear that obtaining a feasible design would be difficult with the CPS algorithm. Consequently, the problem formulation was changed to one that would be more amenable to gradient-based methods. In addition, more design freedom was added by introducing four new design parameters into the optimization problem. This new formulation of the optimization problem was

Table 2: The sequence of optimization results for the electronics package.

Design	Mass (kg)	Violations	Worst SM
Nominal	11.143	8	-0.4797
CPS	11.747	4	-0.2141
NPSOL-SQP and DOT-MMFD	11.739	4	-0.05867
Verified Approx. Opt.	11.997	0	+0.06031

$$\begin{array}{ll} \text{minimize} & M \\ \text{subject to} & SM_i \leq SM_{target}, \text{for } i = 1, \ldots, 55, \\ & (x_n)_{LB} \leq x_n \leq (x_n)_{UB}, \end{array} \tag{4}$$

where $SM_{target} = 0$, and $n = 1, \ldots, 9$. This formulation reduces nonsmoothness by eliminating the possibility of switching between the minimum margin function, as it allows the optimizer to track each margin function independently in the constraints. This does not totally eliminate all sources of nonsmoothness, however, since switching in space and time of the critical response within the finite element block covered by a single margin function is still possible.

The sequential quadratic programming (SQP) method in NPSOL [15] was configured to run with parallel central finite differencing. For $n = 9$ variables, this gives a total of $2n + 1 = 19$ concurrent function evaluations. Given 10 concurrent Salinas executions on ASCI Red, the 19 jobs could be completed in two passes. NPSOL was able to improve the minimum safety margin from -0.21 (the best CPS results) to -0.15 and reduce the total mass to 4.4% over nominal, using a total of 133 function evaluations. Unfortunately, NPSOL was not able to run more than a few cycles before one of the Salinas jobs hung. This coincided with the expiration of the special allotment of 2560 ASCI Red processors that had been dedicated to this study.

6.1.3 DOT-MMFD Algorithm

The optimization process was continued using 256 ASCI Red processors. However, the decision was made to switch from NPSOL's SQP algorithm to DOT's Modified Method of Feasible Directions (MMFD) [16] algorithm. The reasoning for this switch was that the DOT-MMFD algorithm is designed to find a feasible point, even at the expense of increasing the objective function value. In contrast, the NPSOL-SQP algorithm is an infeasible method that only satisfies the constraints at convergence.

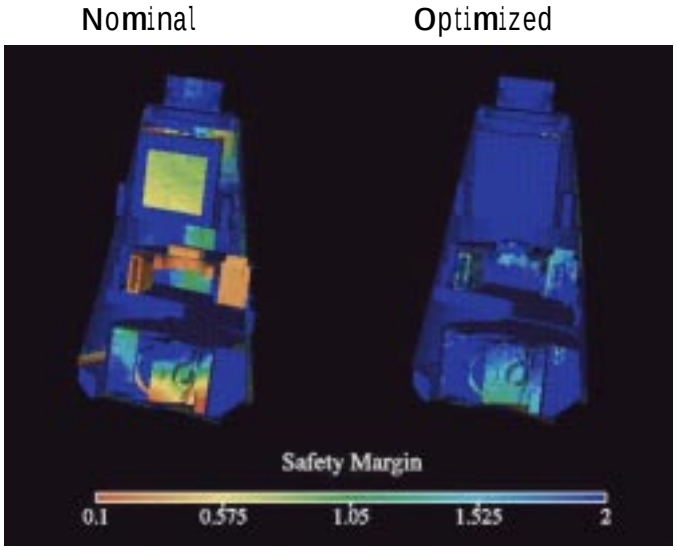

Figure 4: A comparison of safety margin levels in the original (left) and optimized electronics package models. The brighter colors indicate lower safety margins.

In addition, NPSOL's gradient-based line search (in user-supplied gradient mode) is ill-suited for cases that use finite differencing without concurrency in function evaluations.

Starting from the NPSOL-SQP best design point, the DOT-MMFD algorithm improved the worst safety margin value from $SM = -0.15$ to $SM = -0.059$, although it did not find a feasible design point. This DOT-MMFD algorithm used 96 Salinas function evaluations before it was terminated.

Table 2 shows the progression of the optimization results for this study. The CPS algorithm and the two NLP algorithms (NPSOL-SQP and DOT-MMFD) combined to move the infeasible nominal EP design to an improved infeasible design. The worst case safety margin violation had been reduced by about a factor of 10, at a cost of a 5.3% increase in the mass of the EP. Figure 4 compares SM contours at a selected time step which highlights the largest contrast between the nominal and best Phase 1 designs.

At this point in the study, DAKOTA had controlled up to 10 concurrent Salinas jobs, each of which used 256 processors. This use of up to 2560 processors was successful in compressing the duration of Phase 1 to four days. Without the use of parallel computing, equivalent calculations using serial optimization and serial simulation would have required in excess of 10 years to complete.

Additional information became available which motivated the next phase of this study. Some Salinas data from earlier parameter study runs was fully post-processed, and it was discovered that a subset of the safety margin functions exhibited considerable nonsmoothness (Figures 5, 6, and 7). One of these nonsmooth functions was active at the DOT-MMFD solution and was preventing any further progress. Thus, the decision was made to switch to an approximate optimization strategy that used surrogate models to smooth the noisy safety margin constraint functions.

6.2 Phase 2: Optimization using Approximate Models

The approximate optimization (AO) strategy used in this study is a simplified version of the sequential approximate optimization strategy described in [17]. This AO strategy is divided into the following steps: (1) move limit (bounds) selection, (2) data sampling, (3) surface fitting to produce surrogate models, (4) optimization using the surrogate models, and (5) validation of the optima predicted in Step (4). The steps of the AO strategy are described below.

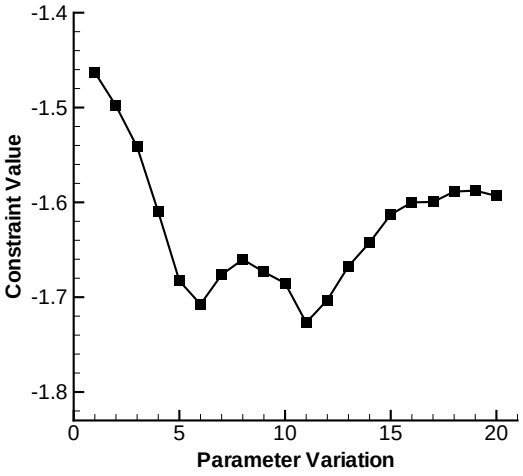

Figure 5: Nonsmooth variations for constraint 30.

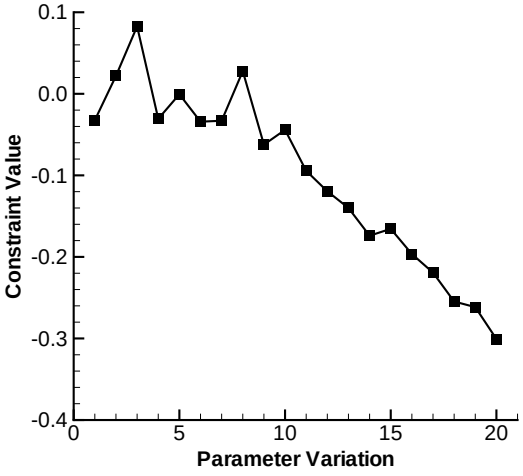

Figure 6: Nonsmooth variations for constraint 53.

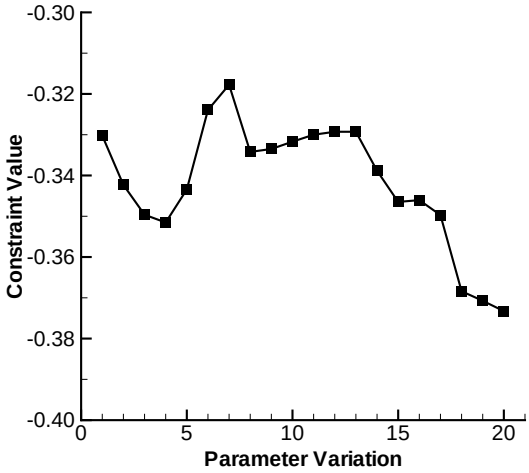

Figure 7: Nonsmooth variations for constraint 56.

6.2.1 Move Limits

The best set of design variables found using DOT-MMFD served as the starting design for the approximate optimization phase. An analysis of the previous optimization data showed that two of the variables did not strongly interact with the optimizer. Thus, these two variables were converted to constants, each having the optimal value obtained from the DOT-MMFD results. The upper and lower bounds on each of the remaining seven variables were reduced to between 18% and 43% of the original bounds based on engineering judgment and the desire to balance the needs of sufficient design freedom and sufficient sampling density. For the remainder of this report, these reduced bounds are referred to as the *move limits* of the approximate optimization.

6.2.2 Latin Hypercube Sampling

Next, the Latin hypercube sampling (LHS) method [18] provided by the DDACE package [19] within DAKOTA was used to generate 200 independent sample locations within the move limits. The 200 samples created by the LHS method correspond to unique EP designs. Salinas was used to evaluate as many of these EP designs as possible using the remainder of the computational budget devoted to this project. This Salinas/DAKOTA calculation again used two-level parallel computing, with four concurrent Salinas jobs, each of which used 256 processors (1024 total processors).

Unfortunately, only 104 of the LHS design points were evaluated during the allocated ASCI Red computer time. While the 104 samples did not compromise a true LHS data set, there were a sufficient number of samples for use

in the approximate optimization phase of this study. One check of the LHS data set involved a preliminary statistical analysis to assess the distribution of the 104 samples in the design space. This analysis did not indicate any correlation or bias among the samples that would have rendered the Latin hypercube samples unusable. Also, the set of 104 samples was sufficient to overfit a 7-dimensional quadratic polynomial (having 36 terms) by almost a factor of three. It is a good rule of thumb to overfit polynomial models whenever possible.

6.2.3 Surrogate Model Construction

The version of DAKOTA used in this study employed four surrogate modeling techniques. These were: (1) kriging spatial interpolation [20, 21]; (2) quadratic polynomial regression (QuadPoly) [22]; (3) multivariate adaptive regression splines (MARS) [23]; and (4) stochastic layered perceptron artificial neural networks (ANN) [24].

The kriging, MARS, and ANN methods do not assume a particular trend in the data. That is, these three surrogate modeling methods can capture arbitrary variations in a given data set. In contrast, the quadratic polynomial regression assumes that the data trends can be modeled using second-order functions. Thus, while all of these surrogate models provide a smooth functional form that is amenable to gradient-based optimization, the QuadPoly surrogate models enforce additional smoothing by nature of the assumed quadratic form.

6.2.4 Optimization with Surrogate Models

Post-processing on each of the 104 Salinas jobs yielded a set of mass and safety margin data. This data was input into DAKOTA in order to build 56 separate surrogate models which define the functional relationships between the objective and constraint functions (mass and 55 safety margins) and the seven EP design parameters. These surrogate models were then used in the optimization problem in place of the Salinas simulations. In this case, multiple optimizations could be performed using the surrogates at very low cost. The drawback is that the surrogate models can be inaccurate, particularly if the optimizer pushes the EP design near the move limit boundaries, where the surrogate models begin to extrapolate the data trends.

The first surrogate model type used in this study was quadratic polynomial regression. That is, the problem defined in Equation 4 was solved using QuadPoly surrogate models for mass and each of 55 constraints. For the initial approximate optimization case, the value of $\mathrm{SM}_{\mathrm{target}}$ in Equation 4 was set to -0.05. Since these surrogate models allow for very inexpensive evaluations, several Monte Carlo sampling studies were performed in order to identify good starting points for the gradient-based optimizations (even though each function is unimodal, their intersections can produce multiple constrained minima). The bound constraints for these gradient-based optimizations were identical to the move limit bounds used in the surrogate model construction. Next, $\mathrm{SM}_{\mathrm{target}}$ was increased to 0.0 and the optimization was performed again. This sequence was continued with $\mathrm{SM}_{\mathrm{target}}$ values of 0.05, 0.10, and 0.15. This was done to generate the mass versus safety margin tradeoff plot shown in Figure 8.

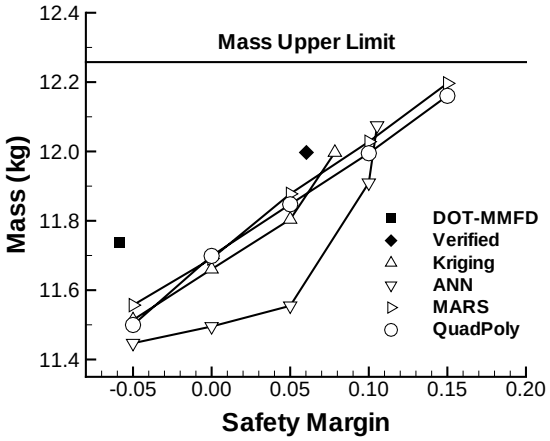

Figure 8: Mass vs. safety margin tradeoff curves generated using various surrogate model types.

A similar sequence of approximate optimizations was performed for each of the other three surrogate model types: kriging, MARS, and ANN. In cases where the safety margin targets were not met, the target was reduced in an iterative fashion until a final maximized safety margin for the surrogate model was achieved. The EP mass versus safety margin tradeoff curves for these surrogate models types also are shown in Figure 8.

There are several interesting items to note about the trends in Figure 8. First, the ANN curve does not follow the same trends as the other three methods. This prompted an examination of the ANN algorithm in DAKOTA, and attempts to further test the ANN algorithm are currently underway. Second, the kriging and ANN tradeoff curves show a kink that results in an increased slope in mass versus SM. This behavior was traced to the optimizer bumping up against one or more of the move limit bounds.

6.2.5 Validation of Approximate Optima

The final step in the entire optimization process was to run Salinas validation analyses for the EP designs identified in the Phase 2 approximate optimization. The best agreement between a predicted EP optimum and its corre-

sponding Salinas validation data occurred for one of optima obtained using the kriging surrogate models. In this case, the actual mass was predicted very accurately (actual and predicted both 11.997 kg), and the actual worst case safety margin value was +0.060 (predicted to be +0.078). The mass and worst case safety margin data for this validated EP design are listed in Table 2.

However, not all of the approximate optima were in such good agreement with the Salinas validation data. In some cases the approximate optima had predicted a positive worst case safety margin, whereas the Salinas validation analysis yielded a negative worst case safety margin. This occurrence underscores the need for validation analyses whenever any type of ad hoc approximate optimization algorithm is employed.

One possible explanation for the successful results obtained with the kriging surrogate models stems from the different manner in which kriging extrapolates data as compared to the other surrogate model types. That is, the kriging model decays to the mean of the data values when used to extrapolate far away from the sample sites. In contrast, the other three surrogate model types extrapolate using the slope of the data, i.e., trends that usually go to $\pm$ infinity far away from the sample sites. See Reference [21] for more information on the extrapolation trends in kriging versus polynomial regression.

Had sufficient computational resources been available, this process would have been continued using a traditional trust-region sequential approximate optimization strategy [17] with additional rounds of sampling, fitting, optimizing, and validating. This would mitigate the validation errors observed previously when only a single approximate optimization cycle is performed.

7 Summary and Conclusions

This paper presents the results of a high-fidelity electronics package design study using a massively parallel structural dynamics code and a multilevel parallel optimization framework.

From the applications perspective, this study demonstrates the utility of having a toolbox of algorithms from which to tailor the optimization procedure as experience with a particular application increases. Through the combination of nongradient, gradient, and approximate optimization methods, the electronics package design was improved from an infeasible design which violated response allowables by a factor of two to a completely feasible design with positive design margins, while still remaining within strict mass targets. In addition, a tradeoff curve of mass versus safety margin was developed to facilitate the design decision process.

From the parallel computing perspective, this paper demonstrates the effectiveness of massively parallel computing in reducing the time to solve an actual engineering design problem. During one portion of the EP design optimization process, a series of studies employed up to 2560 processors in a combination of coarse-grained and fine-grained parallel processing. These studies were completed in four days, where equivalent calculations on a single desktop computer would require in excess of 10 years. Clearly, the redesign of the EP, using a 500,000 degree-of-freedom finite element model, could not have been accomplished without the use of massively parallel computing.

While certain aspects of current-generation custom supercomputers do not yet lend themselves to routine studies of this type, several directions for improvement have been identified. It is expected that advances in optimization and supporting parallel software will be successful in making high-fidelity studies of this type a standard component of modeling and simulation activities in the Department of Energy complex.

References

[1] Reese, G.M., Bhardwaj, M.K., Driessen, B., Alvin, K.F., and Day, D., "Salinas - An Implicit Finite Element Structural Dynamics Code Developed for Massively Parallel Platforms," AIAA paper 2000-1651, 41st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Atlanta, GA, April 3-6, 2000.

[2] Tuminaro, R.S., Heroux, M., Hutchison, S.A., and Shadid, J.N., "Official Aztec User's Guide: Version 2.1," Sandia Technical Report SAND99-8801J, Sandia National Laboratories, Albuquerque, NM, December 1999.

[3] Farhat, C. and Roux, F., "An unconventional domain decomposition method for an efficient parallel solution of large-scale finite element systems," *SIAM J. Sci. Statist. Comput.*, Vol. 13, No. 1, pp. 379-396, 1992.

[4] *ANSYS User Guide*, ANSYS Inc., Southpointe 275 Technology Drive, Canonsburg, PA.

[5] Day, D., "A Basic Parallel Sparse Eigensolver for Structural Dynamics," Sandia Technical Report SAND98-0410C, Sandia National Laboratories, Albuquerque, NM, 1998.

[6] Maschhoff, K.J., and Sorensen, D.C., "PARPACK: An Efficient Portable Large Scale Eigenvalue Package for Distributed Memory Parallel Architectures," *Applied Parallel Computing in Industrial Problems*

and Optimization, Lecture Notes in Computer Science (eds. J. Wasniewski, J. Dongarra, K. Madsen, and D. Olesen), Vol. 1184, Springer–Verlag, Berlin, 1996.

[7] Eldred, M.S., Bohnhoff, W.J., and Hart, W.E., "DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Sensitivity Analysis, and Uncertainty Quantification," Sandia Technical Report SAND01-XXXX (in preparation), Sandia National Laboratories, Albuquerque, NM. (Draft available online from: http://endo.sandia.gov/DAKOTA/papers /Dakota_online.pdf)

[8] Eldred, M.S., Hart, W.E., Schimel, B.D., and van Bloemen Waanders, B.G., "Multilevel Parallelism for Optimization on MP Computers: Theory and Experiment," paper AIAA-2000-4818 in the *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, September 6-8, 2000.

[9] Greenberg, D.S., Brightwell, R., Fisk, L.A., Maccabe, A.B., and Riesen, R.E., "A System Software Architecture for High-End Computing," *Proceedings of Supercomputing 97*, San Jose, CA, 1997.

[10] Mattson, T.G. and Henry, G., "The ASCI Option Red Supercomputer," Intel Supercomputer Users Group, Thirteenth Annual Conference, Albuquerque, NM, June 11-13, 1997. (Available online from: http://www.cs.sandia.gov/ISUG97/ papers/Mattson/OVERVIEW.html)

[11] Snir, M., Otto, S., Huss-Lederman, S., Walker, D., and Dongarra, J., *MPI: The Complete Reference*, MIT Press, Cambridge, MA 1996.

[12] Cougar Operating System (online document), Sandia National Laboratories, Albuquerque, NM. (Available online from: http://www.sandia.gov/ASCI/TFLOP/ Architecture2.1.html)

[13] Hart, W.E., "Coordinate pattern search methods," Sandia Technical Report SAND01-XXXX (in preparation) Sandia National Laboratories, Albuquerque, NM.

[14] Hart, W. E., "SGOPT, A C++ Library of Stochastic Global Optimization Algorithms," Sandia Technical Report SAND01-XXXX, (in preparation) Sandia National Laboratories, Albuquerque, NM.

[15] Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H., "Users Guide for NPSOL (Version 4.0): A Fortran Package for Nonlinear Programming," System Optimization Laboratory, TR SOL-86-2, Stanford University, Stanford, CA, 1986.

[16] *DOT Users Manual, Version 4.20*, Vanderplaats Research and Development, Inc., Colorado Springs, 1995.

[17] Giunta, A.A., and Eldred, M.S., "Implementation of a Trust Region Model Management Strategy in the DAKOTA Optimization Toolkit," paper AIAA-2000-4935 in *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, September 6-8, 2000.

[18] McKay, M. D., Beckman, R. J., and Conover, W. J., "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, Vol. 21, No. 2, 1979, pp. 239-245.

[19] Tong, C.H., and Meza, J.C., "DDACE: A Distributed Object-Oriented Software with Multiple Samplings for the Design and Analysis of Computer Experiments," Sandia Technical Report SAND01-XXXX, (in preparation) Sandia National Laboratories, Albuquerque, NM. (Available online from: http://csmr.ca.sandia.gov/projects/ddace.html)

[20] Cressie, N., *Statistics for Spatial Data*, John Wiley and Sons, Inc., New York, 1991, pp. 1-26.

[21] Giunta, A. A., and Watson, L. T., "A Comparison of Approximation Modeling Techniques: Polynomial Versus Interpolating Models," paper AIAA-98-4758 in *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, 1998, pp. 392-404.

[22] Myers R. H., and Montgomery, D. C., *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, Wiley, New York, 1995, pp. 79-123.

[23] Friedman, J. H., "Multivariate Adaptive Regression Splines," *The Annals of Statistics*, 1990. (also published as Tech Report PUB-4960, Stanford Linear Accelerator Center, 1990)

[24] Zimmerman, D., "Genetic Algorithms for Navigating Expensive and Complex Design Spaces," Final Research Report prepared for Sandia National Laboratories (technical contact M. Eldred), 1996.